This is the accepted version of a paper published in *International Journal of Computer Mathematics*. This paper has been peer-reviewed but does not include the final publisher proof-corrections or journal pagination.

Citation for the original published paper (version of record):

BENCHOP—The BENCHmarking project in Option Pricing.
http://dx.doi.org/10.1080/00207160.2015.1072172

Access to the published version may require subscription.

N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-260897
The aim of the BENCHOP project is to provide the finance community with a set of common benchmark problems that can be used both for comparisons between methods and for evaluation of new methods. Furthermore, in order to facilitate comparisons, MATLAB
implementations of a wide range of existing methods for each benchmark problem will be made available through the BENCHOP web site www.it.uu.se/research/project/compfin/benchop.

We also aim for BENCHOP to serve as a takeoff for future development of methods in option pricing. We expect future papers in the field to use the BENCHOP codes and problems to evaluate performance. In this way, we can contribute to a more uniform and comparable evaluation of the relative strengths and weaknesses of proposed methods.

The benchmark problems have been chosen in such a way as to be relevant both for practitioners and researchers. They should also be possible to implement with a reasonable effort. We have selected problems with respect to a number of features that may be numerically challenging. These are early exercise properties, barriers, discrete dividends, local volatility, stochastic volatility, jump diffusion, and two underlying assets. We have also included evaluation of hedging parameters in one of the problems, as this adds additional difficulties.

In this paper, we present the benchmark problems with sufficient detail so that other people can solve them in the future. We also provide analytical solutions where such are available or methods for computing accurate reference solutions otherwise. Each problem is solved using MATLAB implementations of a number of already existing numerical methods, and timing results are provided as well as error plots. For details of the methods, we refer to the original papers and additional notes at the BENCHOP web site. The codes are not fully optimized, and the numerical results should not be interpreted as competition scores. We rather see it as a synoptical exposition of the qualities of the different methods.

In Section 2 we state and motivate the benchmark problems while the numerical methods are briefly presented in Section 3. Section 4 is dedicated to the presentation of the numerical results and finally in Section 5 we discuss the results. In Appendix A we present how the reference values are computed for the different problems and in Appendix B we discuss how the local volatility surface is computed for one of the problems.

2. Benchmark problems

In this section we state each of the six benchmark problems. In the mathematical formulations, we let $S$ represent the actual (stochastic) asset price realization, whereas $s$ is the asset price variable in the PDE formulation of the problem, $t$ is the time (with $t = 0$ representing today), $r$ is the risk free interest rate, $\sigma$ is the volatility, $W$ is a Wiener process, $u$ is the option price as a function of $s$, $K$ is the strike price, and $T$ is the time of maturity. The payoff function $\phi(s)$ is the value of the option at time $T$. In Problem 4, $V$ is the stochastic variance variable, and $v$ is the variance value in the PDE-formulation.

In practice, the asset value today, $S_0$, is a known quantity, while the strike price $K$ can take on different values. In the benchmark problem descriptions below we have chosen to fix all parameters, and then solve for different values of $S_0$ to simulate pricing of options that are ‘in the money’, ‘at the money’ and ‘out of the money’. The initial values are not given in the problem descriptions, but for each table and figure in the numerical results section, the values that were used are listed.
2.1 Problem 1: The Black–Scholes–Merton model for one underlying asset

The celebrated Black–Scholes–Merton [4, 39] option pricing model, developed in the early 70’s, is arguably the most successful quantitative model ever introduced in social sciences, even initiating the new field of Financial Engineering, which occupies thousands of researchers in financial institutions and universities across the world.

A key property of the model is that by building on so-called no-arbitrage arguments, it allows the price of plain vanilla call and put options to be calculated using variables that are either directly observable or can be easily estimated. The model is still widely used as a benchmark, although more advanced models have been developed over the years to take into account real-world features of asset prices dynamics, such as jumps and stochastic volatility (see below).

The Black–Scholes–Merton model has the advantage that closed form solutions exist for prices, as well as for hedging parameters, for some types of options. It has therefore been extensively used to test numerical methods that are then applied to more advanced problems. The computation of the hedging parameters (Greeks) is included in this benchmark problem as they are of significant practical interest and can be expensive and/or difficult to compute for some numerical methods.

Mathematical formulation

\[ \text{SDE-setting: } dS = rSdt + \sigma SdW. \] (1)

\[ \text{PDE-setting: } \frac{\partial u}{\partial t} + \frac{1}{2} \sigma^2 s^2 \frac{\partial^2 u}{\partial s^2} + rs \frac{\partial u}{\partial s} - ru = 0. \] (2)

Deliverables

The pricing problem should be solved for three types of options; a) a European call option, b) an American put option, and c) a barrier option. For the European option also the most common hedging parameters \( \Delta = \frac{\partial u}{\partial s}, \Gamma = \frac{\partial^2 u}{\partial s^2} \) and \( \nu = \frac{\partial u}{\partial \sigma} \) should be computed.

Parameter and problem specifications

For this problem we have two sets of model parameters, representing less and more numerically challenging situations, respectively.

Standard parameters: \( \sigma = 0.15, \ r = 0.03, \ T = 1.0, \ \text{and} \ K = 100. \) (3)

Challenging parameters: \( \sigma = 0.01, \ r = 0.10, \ T = 0.25, \ \text{and} \ K = 100. \) (4)

The three types of options are characterized by their exercise properties and payoff functions.
a) European call: \[ \phi(s) = \max(s - K, 0). \]
b) American put: \[ \phi(s) = \max(K - s, 0), \]
    \[ u(s, t) \geq \phi(s), \quad 0 \leq t \leq T. \]
c) Barrier call up-and-out: \[ \phi(s) = \begin{cases} 
    \max(s - K, 0), & 0 \leq s < B \\
    0, & s \geq B 
\end{cases}, \quad B = 1.25K. \]

### 2.2 Problem 2: The Black–Scholes–Merton model with discrete dividends

A shortcoming of the classical Black-Scholes formula is that it is only valid if the underlying stock does not pay dividends, invalidating the approach for many stocks in practice. In some special cases, e.g., when dividend yields are constant and paid continuously over time, closed form solutions can be derived for dividend paying stocks too, see [39]. Usually, however, numerical methods are needed to calculate the option’s value.

In practice, dividends are paid at discrete points in time, and the size of the dividend payments depends on the performance of the firm. For example, a firm whose performance has been poor may be capital constrained and therefore choose not to make a dividend payment, as may a company that needs its capital for a new investment opportunity. Fairly advanced stochastic modeling may therefore be needed in practice to capture the dividend dynamics of a company. In numerical tests, it is common to abstract away from these issues and simply assume that the firm makes discrete proportional dividend payments (i.e., has a constant dividend yield).

#### Mathematical formulation

**SDE-setting:**
\[ dS = rSdt + \sigma SdW - \delta(t - \tau)D S dt. \]  \hspace{1cm} (5)

**PDE-setting:**
\[ \frac{\partial u}{\partial t} + \frac{1}{2} \sigma^2 s^2 \frac{\partial^2 u}{\partial s^2} + rs \frac{\partial u}{\partial s} - ru = 0, \]  \hspace{1cm} (6)

In the SDE case, the (single) dividend at time \( \tau \) enters explicitly, whereas in the PDE case, it is implicitly taken into account by enforcing
\[ u(s, \tau^-) = u(s(1 - D), \tau^+). \]  \hspace{1cm} (7)

#### Deliverables

Prices should be computed for a) a European call option and b) an American call option.

#### Parameter and problem specifications

The dividend is defined by \( \tau = 0.4 \) and \( D = 0.03 \). We use the standard parameters (3) except for the expiration time, set to \( T = 0.5 \), together with standard payoff function \( \phi(s) = \max(s - K, 0) \) and European and American exercise properties respectively, see Problem 1.
2.3 Problem 3: The Black–Scholes–Merton model with local volatility

As mentioned earlier, the Black–Scholes–Merton model with a constant volatility does not reproduce market prices very well in practice. One discrepancy is the so-called volatility smile (which after the October 1987 crash is known to have turned into a smirk). If the implied volatility—the volatility in the Black–Scholes–Merton model that is consistent with the observed option price—is calculated for several options with the same exercise date but different strike prices, all options should under the classical assumptions of Black–Scholes–Merton have the same implied volatility. Instead, when plotted against the different strike prices, the curve is usually that of a U-shaped smile (or an L-shaped smirk).

As discussed in [10], an approach to address this discrepancy between model and data is to assume local volatility, i.e., to allow the volatility of the underlying asset to depend instantaneously on the stock price \( s \), and time \( t \), generating a whole volatility surface. It is shown in [10] how to reverse engineer such a volatility surface from observed option prices.

Given a volatility surface, the general Black–Scholes–Merton no-arbitrage approach can be used to derive the option price, although closed form solutions will typically no longer exist. We provide two volatility surfaces with different properties in order to see how the numerical methods handle such variable volatility coefficients.

Mathematical formulation

The equations for the option price are identical to (1) and (2) except that here \( \sigma = \sigma(s, t) \).

Deliverables

The price for a European call option should be computed in each case.

Parameter and problem specifications

The first local volatility surface is given by an explicit function

\[
\sigma_I(s, t) = 0.15 + 0.15(0.5 + 2t) \frac{(s/100 - 1.2)^2}{(s/100)^2 + 1.44}.
\]  

The second local volatility surface \( \sigma_{II}(s, t) \) is based on market data and does not have an explicit form. The local surface is computed from a parametrization of the implied volatility data. The exact steps in the computation and the specific parametrization are given in Appendix B.

In both cases, we use \( K = 100 \) and \( r = 0.03 \), but the expiration times are different, with \( T = 1 \) for \( \sigma_I \), and \( T = 0.5 \) for \( \sigma_{II} \). The payoff function for a European call option is as before \( \phi(s) = \max(s - K, 0) \).

2.4 Problem 4: The Heston model for one underlying asset

The local volatility model allows for perfect matching of prices of European-style options but, just like the Black–Scholes–Merton model, also has its weaknesses. It does not perform very well for path dependent options and, moreover, there is clear evidence that
in practice the volatility of asset prices is in itself random, beyond what can be simply be described as a function of time and underlying strike price [9, 32, 45]. The Heston model [20] assumes that in addition to the risk-factor that drives the value of the underlying asset, there is another risk-factor that determines the underlying’s instantaneous variance, $V$. The PDE formulation of the model is therefore two-dimensional. Note that in contrast to the previous models, the market in Heston’s model is incomplete, and therefore additional assumptions about the market price of volatility risk are needed to determine the option price. The specific assumptions in [20] leads to the model below.

**Mathematical formulation**

SDE-setting:

$$dS = rSdt + \sqrt{V}SdW_1,$$
$$dV = \kappa(\theta - V)dt + \sigma\sqrt{V}dW_2,$$  \hspace{1cm} (9)

where $W_1$ and $W_2$ have correlation $\rho$.

PDE-setting:

$$\frac{\partial u}{\partial t} + \frac{1}{2}v^2s^2\frac{\partial^2 u}{\partial s^2} + \rho \sigma v^2s \frac{\partial^2 u}{\partial s \partial v} + \frac{1}{2} \sigma^2 v^2 \frac{\partial^2 u}{\partial v^2} + rs \frac{\partial u}{\partial s} + \kappa(\theta - v) \frac{\partial u}{\partial v} - ru = 0.$$  \hspace{1cm} (10)

**Deliverables**

The price for a European call option should be computed.

**Parameter and problem specifications**

The model parameters are here given by $r = 0.03$, $\kappa = 2$, $\theta = 0.0225$, $\sigma = 0.25$, $\rho = -0.5$, $K = 100$, and $T = 1$. The payoff function for the European call option is $\phi(s,v) = \max(s - K, 0)$. With these parameters, the Feller condition is satisfied.

2.5 **Problem 5: The Merton jump diffusion model for one underlying asset**

The Merton model [40] addresses another difference between real world asset price dynamics and the (local volatility) Black–Scholes–Merton model. What was identified early on, is that stock prices occasionally experience dramatic movements over very short time periods, i.e., they sometimes ‘jump’. Such jumps make return distributions heavier-tailed than for pure diffusion processes, also in line with empirical observations and, as in the Heston model, causes the market to be incomplete, necessitating additional assumptions to price the option. The assumption used in [40] is that the underlying stock price follows a jump-diffusion process, where there is no risk-premium associated with jump risk. Under these conditions, the option price can be computed from a Partial-Integro Differential Equation (PIDE).
**Mathematical formulation**

SDE-setting:

\[
dS = (r - \lambda \xi)Sdt + \sigma SdW + SdQ,
\]

where \( Q \) is a compound Poisson process with intensity \( \lambda > 0 \) and jump ratios that are log-normally distributed as \( p(y) = \frac{1}{\sqrt{2\pi} \delta} e^{-\frac{(\log y - \gamma)^2}{2\delta^2}} \) [50].

PIDE-setting:

\[
\frac{\partial u}{\partial t} + \frac{1}{2} \sigma^2 s^2 \frac{\partial^2 u}{\partial s^2} + (r - \lambda \xi) s \frac{\partial u}{\partial s} - (r + \lambda) u + \lambda \int_0^\infty u(sy, \tau) p(y) dy = 0.
\]

**Deliverables**

The price for a European call option should be computed.

**Parameter and problem specifications**

The parameters to use are \( r = 0.03, \lambda = 0.4, \gamma = -0.5, \delta = 0.4, \xi = e^{\gamma + \delta^2/2} - 1, \sigma = 0.15, K = 100, \) and \( T = 1. \) The payoff function is \( \phi(s) = \max(s - K, 0). \)

2.6 **Problem 6: The Black–Scholes–Merton model for two underlying assets**

As an example of an option with more than one underlying, we use a spread option, which for \( K = 0 \) is called a Margrabe option [38]. This classic rainbow option has a payoff function that depends on two underlying assets, so the option price dynamic therefore depends on two risk-factors (as long as the two stocks' returns are not perfectly correlated). In contrast to the Heston and Merton models, the model is still within the class of complete market models, and the option price is therefore completely determined without further assumptions. The reason that the market is complete in this case, in contrast to the other multi risk-factor models we have introduced, is that two underlying risky assets may be used in the formation of a hedging portfolio, whereas only one such asset is available with the Heston and Merton Models.

**Mathematical formulation**

SDE-setting:

\[
\begin{align*}
   dS_1 &= rS_1 dt + \sigma_1 S_1 dW_1, \\
   dS_2 &= rS_2 dt + \sigma_2 S_2 dW_2,
\end{align*}
\]

where \( W_1 \) and \( W_2 \) have correlation \( \rho. \)
PDE-setting:
\[
\frac{\partial u}{\partial t} + \frac{1}{2} \sigma_1^2 s_1^2 \frac{\partial^2 u}{\partial s_1^2} + \rho \sigma_1 \sigma_2 s_1 s_2 \frac{\partial^2 u}{\partial s_1 \partial s_2} + \frac{1}{2} \sigma_2^2 s_2^2 \frac{\partial^2 u}{\partial s_2^2} + rs_1 \frac{\partial u}{\partial s_1} + rs_2 \frac{\partial u}{\partial s_2} - ru = 0. \tag{14}
\]

**Deliverables**
The price for a European spread call option should be computed.

**Parameter and problem specifications**
The model parameters to use are \( r = 0.03, \sigma_1 = \sigma_2 = 0.15, \rho = 0.5, K = 0, \) and \( T = 1. \)
The payoff function for the European call spread option is \( \phi(s_1, s_2) = \max(s_1 - s_2 - K, 0). \)

3. **Numerical methods**

In Table 1 we display all methods that we have used. We also provide references to the original papers describing the methods. More information about the particular implementations used here can be found at [www.it.uu.se/research/project/compfin/benchop](http://www.it.uu.se/research/project/compfin/benchop).

4. **Numerical results**

For each benchmark problem, we have decided on three (or five) evaluation points \( s_i \) (or \( (s_i, s_j) \)). Each method must be tuned such that it delivers a solution \( u(s_i) \) with a relative error less than \( 10^{-4} \) in these points. We have not put any restrictions on the error in the rest of the domain. Due to this freedom, some codes have been tuned to narrowly target these points, while others (sometimes automatically) are tuned to achieve an evenly distributed error. Then the codes are run (on the same computer system) and the execution times are recorded. Each code is run 4 times, and the execution time reported in the tables below is the average of the last three runs. This is because the first time a MATLAB script is executed in a session it takes a bit longer.

In the tables, we also show the approximate number of correct digits \( p \) in the result. This quantity is computed as \( p = \lceil -\log_{10} e_r \rceil \), where \( e_r \) is the maximum relative error and \( \lceil \cdot \rceil \) indicates rounding. The maximum relative error is computed as
\[
e_r = \max_i \left| \frac{u(s_i) - u^{ref}(s_i)}{u^{ref}(s_i)} \right|.
\]

For the Monte Carlo methods where the error is not deterministic, the errors are averaged over the different runs. In some cases, a method was not able to reach a relative error of \( 10^{-4} \) within reasonable time (1 hour), but a lower target \( 10^{-3} \) was attainable. These results are marked with a * in the tables. The execution time we report in the tables for Monte Carlo methods is the time to compute the result for one evaluation point, whereas for other methods it is the time to compute the result for all evaluation points.
Table 1. List of methods used with abbreviations, marker symbol used in figures, and references

<table>
<thead>
<tr>
<th>Abbr.</th>
<th>Symbol</th>
<th>Method</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC</td>
<td>⬠</td>
<td>Monte Carlo with Euler-Maruyama in time</td>
<td>[16]</td>
</tr>
<tr>
<td>MC-S</td>
<td>△</td>
<td>Monte Carlo with analytical solution / Euler-Maruyama / quadratic scheme in time and stratified sampling</td>
<td>[16], [17], [42], [50], [36], [2]</td>
</tr>
<tr>
<td>QMC-S</td>
<td>×</td>
<td>Quasi Monte Carlo with analytical solution / Euler-Maruyama / quadratic scheme in time, stratified sampling, and precomputed quasi random numbers</td>
<td>[16], [17], [42], [50], [36], [2], [21], [44]</td>
</tr>
<tr>
<td>FFT</td>
<td>□</td>
<td>Fourier method with FFTs</td>
<td>[6], [33], [31]</td>
</tr>
<tr>
<td>FGL</td>
<td>⬠</td>
<td>Fourier method with Gauss-Laguerre quadrature</td>
<td>[1, Page 890], [7], [18], [31], 34, Section 2.1–2.2, [35]</td>
</tr>
<tr>
<td>COS</td>
<td>◦</td>
<td>Fourier method based on Fourier cosine series and the characteristic function.</td>
<td>[11], [12], [52], [53]</td>
</tr>
<tr>
<td>FD</td>
<td></td>
<td>Finite differences on uniform grids with Rannacher smoothed CN in time</td>
<td>[64, Chapter 78], [51], [23], [59]</td>
</tr>
<tr>
<td>FD-NU</td>
<td>⬠</td>
<td>Finite differences on quadratically refined grids with Rannacher smoothed CN / IMEX-CNAB in time</td>
<td>[22], [49], [51], [55], [56]</td>
</tr>
<tr>
<td>FD-AD</td>
<td>⬠</td>
<td>Adaptive finite differences with discontinuous Galerkin / BDF-2 in time</td>
<td>[46], [47], [26], [62], [22], [19]</td>
</tr>
<tr>
<td>RBF</td>
<td>⬠</td>
<td>Global radial basis functions with non-uniform nodes and BDF-2 in time</td>
<td>[48], [19]</td>
</tr>
<tr>
<td>RBF-FD</td>
<td>⬠</td>
<td>Radial basis functions generated finite differences with BDF-2 in time</td>
<td>[41], [63], [58], [19], [13], [14], [65]</td>
</tr>
<tr>
<td>RBF-PUM</td>
<td>△</td>
<td>Radial basis functions partition of unity method with BDF-2 in time</td>
<td>[54], [57], [19]</td>
</tr>
<tr>
<td>RBF-LSML</td>
<td>⬠</td>
<td>Least-squares multi-level radial basis functions with BDF-2 in time</td>
<td>[28], [27], [19]</td>
</tr>
<tr>
<td>RBF-AD</td>
<td>⬠</td>
<td>Adaptive RBFs with CN in time</td>
<td>[43], [8], [24]</td>
</tr>
<tr>
<td>RBF-MLT</td>
<td>△</td>
<td>Multi-level radial basis functions treating time as a spatial dimension</td>
<td>[25], [30], [60]</td>
</tr>
</tbody>
</table>

In order to see how the errors behave away from the evaluation points, solutions are also plotted for a range of values. The figures below show the absolute errors evaluated at the integer values between \( s = 60 \) and \( s = 160 \). No figure is shown for the second local volatility case in Problem 3, because there the local volatility result is only valid for a particular \( S_0 \), not over a range. The vertical axis range in the figures is adjusted to the values that are plotted, but the lower limit is not allowed to be lower than \( 10^{-20} \). Errors falling below that value are not visible in the figures.

The experiments have been performed on the Tintin cluster at Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX), Uppsala University. The cluster consists of 160 dual AMD Opteron 6220 (Bulldozer) nodes. All codes are implemented (serially) in MATLAB. The names of the respective codes for each problem are indicated by the boldfaced heading over each (group of) plot(s). This generic name is then combined with an acronym for the particular method as for example `BSeuCallUI_RBF.m`. 

9
5. Discussion

Monte Carlo methods. MC methods are easy to implement in any number of dimensions, but the slow convergence rate, $O(1/\sqrt{N})$ for standard MC, makes it computationally expensive to reach the requested tolerance of $10^{-4}$. The most challenging problems for the MC methods were the path dependent options, the hedging parameter $\Gamma$, and the local volatility.

Because MC methods scale linearly with the number of dimensions, they are increas-
Table 4. Problems 2 and 3. Computational time to compute a solution $u$ that has a relative error $< 10^{-4}$ at $t = 0$ and $s = 90, 100, 110$. The numbers within parentheses indicate the approximate number of correct digits in the result. A '-' indicates not implemented, while '×' means implemented, but not accurate.

<table>
<thead>
<tr>
<th>Method</th>
<th>Discrete dividends</th>
<th></th>
<th>Local volatility</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>European call</td>
<td>American call</td>
<td>Smooth</td>
</tr>
<tr>
<td>MC-S</td>
<td>6.0e+01 (3)</td>
<td></td>
<td>×</td>
</tr>
<tr>
<td>QMC-S</td>
<td>1.6e+00 (4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FFT</td>
<td>1.5e-03 (7)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FGL</td>
<td>3.6e-03 (14)</td>
<td>8.1e-02 (6)</td>
<td></td>
</tr>
<tr>
<td>COS</td>
<td>8.8e-04 (5)</td>
<td>2.4e-03 (4)</td>
<td>1.7e-02 (4)</td>
</tr>
<tr>
<td>FD</td>
<td>2.0e-02 (4)</td>
<td>1.5e-02 (4)</td>
<td>2.2e-02 (4)</td>
</tr>
<tr>
<td>FD-NU</td>
<td>1.6e-02 (4)</td>
<td>1.5e-02 (4)</td>
<td>3.7e-02 (4)</td>
</tr>
<tr>
<td>FD-AD</td>
<td>2.1e-02 (4)</td>
<td>2.5e-02 (5)</td>
<td>3.6e-02 (4)</td>
</tr>
<tr>
<td>RBF</td>
<td>2.3e-01 (4)</td>
<td>1.1e-01 (4)</td>
<td>5.5e-02 (4)</td>
</tr>
<tr>
<td>RBF-FD</td>
<td>4.2e-01 (4)</td>
<td>2.7e+00 (4)</td>
<td>2.2e+01 (4)</td>
</tr>
<tr>
<td>RBF-PUM</td>
<td>3.3e-02 (4)</td>
<td>3.0e-02 (4)</td>
<td>1.4e-01 (4)</td>
</tr>
<tr>
<td>RBF-LSML</td>
<td>5.0e-01 (4)</td>
<td>1.2e+00 (4)</td>
<td>1.7e-01 (4)</td>
</tr>
</tbody>
</table>

Table 5. Problems 4, 5, and 6. Computational time to compute a solution $u$ that has a relative error $< 10^{-4}$ at $t = 0$ and $s = 90, 100, 110$ for the Heston and Merton models, and to compute a solution $u$ that has a relative error $< 10^{-4}$ at $t = 0$ and $(s_1, s_2) = (100, 90), (100, 100), (100, 110), (90, 100), (110, 100)$ for the spread option. The numbers within parentheses indicate the approximate number of correct digits in the result. A '-' indicates not implemented, while '×' means implemented, but not accurate.

<table>
<thead>
<tr>
<th>Method</th>
<th>Heston</th>
<th>Merton</th>
<th>Spread</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC-S</td>
<td>×</td>
<td>1.6e+01 (4)</td>
<td>*2.9e+01 (4)</td>
</tr>
<tr>
<td>QMC-S</td>
<td>3.4e-01 (4)</td>
<td>1.8e+00 (5)</td>
<td></td>
</tr>
<tr>
<td>FFT</td>
<td>3.3e-03 (5)</td>
<td>2.1e-03 (5)</td>
<td></td>
</tr>
<tr>
<td>FGL</td>
<td>3.8e-03 (14)</td>
<td>3.1e-03 (13)</td>
<td>3.1e-03 (14)</td>
</tr>
<tr>
<td>COS</td>
<td>3.4e-04 (4)</td>
<td>2.2e-04 (4)</td>
<td>1.5e-03 (4)</td>
</tr>
<tr>
<td>FD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FD-NU</td>
<td>4.3e+00 (4)</td>
<td>1.4e-01 (4)</td>
<td>7.4e+01 (4)</td>
</tr>
<tr>
<td>FD-AD</td>
<td></td>
<td></td>
<td>4.7e+01 (4)</td>
</tr>
<tr>
<td>RBF</td>
<td>1.9e+01 (4)</td>
<td></td>
<td>7.7e+01 (4)</td>
</tr>
<tr>
<td>RBF-FD</td>
<td></td>
<td></td>
<td>2.2e+03 (4)</td>
</tr>
<tr>
<td>RBF-PUM</td>
<td>4.3e+00 (4)</td>
<td></td>
<td>1.3e+01 (5)</td>
</tr>
</tbody>
</table>

ingly competitive in higher dimensions. Furthermore, there are a lot of specialized techniques that can be applied to improve performance. An example of this is the QMC-S method that is comparable to some of the other methods already in one dimension, and the fastest method apart from the Fourier methods for the two-dimensional spread option.

**Fourier methods.** Fourier methods (FM) rely on the availability of the characteristic function (ChF) of the underlying stochastic process. These are available for all problems except Problem 3, local volatility. However, in the recent publication [53], the stochastic process is approximated by a second order weak Taylor scheme, for which there exists an analytic solution for the ChF. This method was applied to the smooth local volatility
function, but could not easily be used for the implied local volatility. Apart from Problem 3, the problems that were most challenging for the FM were the American and Up-and-
out options.

The FM are all very fast and especially the FGL-method is also highly accurate. The fastest FM, the COS method, is the overall fastest method in all cases but two. The competitiveness of the FM is even more pronounced for the two-dimensional problems, the Heston model and the spread option.

*Finite difference methods.* FD methods rely on the use of structured (possibly non-uniform) grids and are straightforward to implement. The FD-NU is the only BENCHOP method that solved all the problems. The computational times are low in all cases, and for two problems FD-NU or FD-AD is the overall fastest method. For the FD methods in general the challenging parameter set in Problem 1 is the most difficult feature to handle.

For Problem 1 the usage of a nonuniform grid, FD-NU and FD-AD, is superior to using a uniform grid (FD) but for Problems 2 and 3, using a uniform grid is sometimes faster. The FD method has not been implemented for the two-dimensional problems, but we believe that FD-NU and FD-AD would be faster than FD in these cases thanks to the possibility of local refinement.

*Radial basis function methods.* RBF methods are flexible with respect to node locations and choice of basis function. This makes it possible to tune the methods to achieve well for particular targets, but it can also be hard to make a good choice. Problem 5, Merton jump diffusion has not been implemented in any RBF method, but this could be done. The problems that are challenging for RBF methods have non-smooth solutions or very sharp gradients, such as American options and the challenging parameter set in Problem 1.

The RBF methods as a group are slower than the FD methods for the one-dimensional problems. However, the results of the fastest RBF method, RBF-PUM, is in the favorable cases of the same order as those of the FD methods. In two dimensions the RBF-PUM method is as fast as or faster than the implemented FD methods. Potentially, the RBF-PUM method will be even more competitive in higher dimensions.

**Acknowledgements**

We would like to thank Institut Mittag-Leffler for supporting the workshop we organized on ”Mathematical and Numerical modeling in Finance” ([http://www.mittag-leffler.se/?q=0609](http://www.mittag-leffler.se/?q=0609)), where the idea for the BENCHOP project was conceived and where the work was initiated.

The computations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) through Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX) under Project snic2014-3-73.
Figure 2. Problem 1a) European call option. Errors in the hedging parameters $\Delta$ (top group), $\Gamma$ (middle group), and $\nu$ (bottom group). For each problem, results for standard parameters are shown to the left, and for challenging parameters to the right. Absolute errors in the hedging parameters for $t = 0$ and $60 \leq s \leq 160$ when the relative error is less than $10^{-4}$ at $t = 0$ and $s = 90, 100, 110$ for standard parameters and $s = 97, 98, 99$ for challenging parameters.
Figure 3. Problem 2 a) European call with dividends (left) and 2b) American call with dividends (right). Error in the solution $u$ for $t = 0$ and $60 \leq s \leq 160$ when the relative error in $u$ is less than $10^{-4}$ at $t = 0$ and $s = 90$, 100, 110.

Figure 4. Problem 3 Local volatility, smooth function (top left), Problem 4 Heston (top right), Problem 5 Merton (bottom left), and Problem 6 Spread option (bottom right). Absolute error in the solution $u$ for $t = 0$ and $60 \leq s \leq 160$ when the relative error in $u$ is less than $10^{-4}$ at $t = 0$ and $s = 90$, 100, 110 for Local volatility and Merton, and for Heston with variance $v = 0.0225$. For the spread option, the error is measured in $(s_1, s_2) = (100, 90), (100, 100), (100, 110), (90, 100), (110, 100)$, and is plotted for $60 \leq s_1 \leq 160$, and $s_2 = 100$. 
Appendix A. The methods used for computing the reference values used in the comparisons

For many of the benchmark problems here, there are analytical or semi-analytical solutions. For these cases, we state the closed form expressions. For the cases lacking analytical solutions, we describe the numerical method that was used for accurate enough computation of a reference solution. MATLAB codes for each problem are available at the BENCHOP web page.

A.1 Problem 1: The Black–Scholes–Merton model for one underlying asset

For the European call option, the closed form expression for the option price is given in [4]

$$
\Pi_{BS}^c(t, S, K, T, r, \sigma^2) = SN(d^+(S/K, T - t)) - Ke^{-r(T-t)}N(d^-(S/K, T - t)),
$$
(A1)

where

$$
d^+(x, y) = \frac{1}{\sigma \sqrt{y}} \left( \log(x) + \left( r + \frac{\sigma^2}{2} \right) y \right),
$$
(A2)

$$
d^-(x, y) = \frac{1}{\sigma \sqrt{y}} \left( \log(x) + \left( r - \frac{\sigma^2}{2} \right) y \right),
$$
(A3)

and where $N(x)$ is the cumulative distribution function for the standard normal distribution. The hedging parameters can be found through differentiation, leading to

$$
\Delta_{BS} = \frac{\partial \Pi_{BS}^c}{\partial S} = N(d^+(S/K, T - t)),
$$
(A4)

$$
\Gamma_{BS}^c = \frac{\partial^2 \Pi_{BS}^c}{\partial S^2} = \frac{\phi(d^+(S/K, T - t))}{S \sqrt{(T-t)\sigma^2}},
$$
(A5)

$$
\nu_{BS}^c = \frac{\partial \Pi_{BS}^c}{\partial \sigma} = S\phi(d^+(S/K, T - t))\sqrt{T-t},
$$
(A6)

where $\phi(x) = (2\pi)^{-1/2} \exp(-x^2/2)$ is the density of the standard normal distribution.

For the American put option, there is no closed form solution. Different analytical and semi-analytical approximations to the location of the early exercise boundary are analyzed and compared in [29]. Here, we use a relation from [5, Theorem 1.1], where it is shown that the American put price can be decomposed into a European put price and the early exercise premium. The general European put price is defined by

$$
\Pi_{BS}^p(t, S, K, T, r, \sigma^2) = -SN(-d^+(S/K, T - t)) + Ke^{-r(T-t)}N(-d^-(S/K, T - t)),
$$
(A7)
The American put price becomes
\[
\Pi_{BS-A}(t, S, K, T, r, \sigma^2) = \Pi_{BS}(t, b_p(t), K, T, r, \sigma^2) \\
+ rK \int_t^T e^{-r(u-t)} N \left( -d^- (b_p(t)/b_p(u), u-t) \right) du,
\]
(A8)

where \( b_p(t), 0 \leq t < T \) is the (unknown) optimal exercise level at time \( t \), and \( b_p(T) = K \).

The location of the exercise boundary is determined by solving the following non-linear integral equation numerically:
\[
\Pi_{BS-A}(t, S, K, T, r, \sigma^2) = K - b_p(t).
\]
(A9)

This equation is solved by an implicit trapezoidal method, where the implicit step due to the monotonicity in \( b_p(t) \) can be found by binary search. This leads to a robust albeit slow method for obtaining the optimal exercise level \( b_p \) on a fine time grid from \( t = T \) to \( t = 0 \). Then, for arbitrary initial stock values satisfying \( S(0) > b_p(0) \), (A8) provides the option value. For \( S(0) \leq b_p(0) \) the value is set to the exercise value \( K - S(0) \).

For the barrier call up–and–out option, a closed form expression for the option price can be found in [3, Theorem 18.12 p. 271].

\[
P^c_{BS-UO}(t, S, K, T, T, r, \sigma^2) = P^c_{BS}(t, S, K, T, r, \sigma^2) - P^c_{BS}(t, B, T, r, \sigma^2) - (B^{2r})_{e^{-r(T-t)}} \left( P^c_{BS}(t, B^2/S, K, T, r, \sigma^2) - P^c_{BS}(t, B^2/S, B, T, r, \sigma^2) \right).
\]
(A10)

A.2 Problem 2: The Black–Scholes–Merton model with discrete dividends

For the European call option with one proportional discrete dividend payment, there is a closed form solution [3, Proposition 16.6 p. 235]. Assuming that the dividend is paid out at time \( \tau \), where \( t < \tau < T \), we have
\[
P^c_{BS-EDD}(t, S, K, T, r, \sigma^2) = P^c_{BS}(t, S(1-D), K, T, r, \sigma^2).
\]
(A11)

Note that the option price is independent of when during the contract period the dividend occurs.

The American Call option with one dividend payment at \( \tau = \alpha T \), where \( 0 < \alpha < 1 \),
can be valuated semi analytically [61].

\[
\Pi_{BS-ADD}(0, S, K, T, r, \sigma, D, \alpha) = \\
(1 - D)SN_2 \left( -d^+ \left( \frac{S}{S_T^*}, \alpha T \right), \frac{(1 - D)S}{K}, T, -\sqrt{\alpha} \right) \\
- Ke^{-rT}N_2 \left( -d^- \left( \frac{S}{S_T^*}, \alpha T \right), \frac{(1 - D)S}{K}, T, -\sqrt{\alpha} \right) \\
+ \Pi_{BS}((1 - \alpha)T, S, S_T^*, T, r, \sigma^2) \\
\]

where \( N_2(x, y, \rho) \) is the cumulative distribution function for the bi-variate normal distribution with zero mean and covariance matrix \( \Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \).

The above formula depends on the unknown variable \( S_T^* \), which can be estimated from the following nonlinear problem:

\[
S_T^* - K = \Pi_{BS}(T - \tau, (1 - D)S_T^*, K, T, r, \sigma^2). 
\]

**A.3 Problem 3: The Black–Scholes–Merton model with local volatility**

For a general local volatility function, there is no closed form solution. The reference values have in this case been computed to high accuracy by a selection of the contributed deterministic methods with different numerical approximations in space, different treatments of the boundary, and different approximations in time. In this way, we can be reasonably certain that numerical bias has been eliminated.

**A.4 Problem 4: The Heston model for one underlying asset**

The (almost exact) Heston price is computed through inverse Fourier transform using the Gauss-Laguerre quadrature method (FGL) with 1000 quadrature points in combination with an optimized choice of integration path in the complex plane stock value for stock value. The integration path is chosen so that is goes through the uniquely given saddle point of the integrand (see [34] and [37]).
A.5 Problem 5: The Merton jump diffusion model for one underlying asset

From [40] we have that the price under Merton jump diffusion is given by a weighted sum of modified Black-Scholes prices

\[
\Pi_{cME}(t, S, K, T, r, \lambda, \gamma, \delta^2, \sigma^2) = \sum_{n=0}^{\infty} e^{-\lambda(T-t)} \left( \frac{\lambda(T-t)^n}{n!} \right) \Pi_{cBSE}(t, (\xi + 1)^ne^{-\lambda(T-t)}S, K, T, \sigma^2 + \delta^2 \frac{n}{T-t}), \quad (A13)
\]

where \( \xi = e^{\gamma+\delta^2/2} - 1 \). By using a few hundred terms, a highly accurate price approximation can be computed.

A.6 Problem 6: The Black–Scholes–Merton model for two underlying assets

For the particular case of \( K = 0 \), a closed form solution for the European call spread option is given in [38] as

\[
\Pi_{cBS-DO}(t, S_1, S_2, T, \sigma_1, \sigma_2, \rho) = \Pi_{cBS}(t, S_1, S_2, T, 0, \sigma_1^2 - 2\rho\sigma_1\sigma_2 + \sigma_2^2). \quad (A14)
\]

Appendix B. The second local volatility function used in Problem 3

The second local volatility function is based on a stochastic volatility inspired (SVI) parametrization [15] of the implied volatility surface. This approach to local volatility is widely used by practitioners, because it is relatively easy to calibrate to data, and there are techniques to eliminate different modes of arbitrage.

B.1 The given SVI parametrization

The global total implied variance surface in terms of the time of maturity \( T \) and the log moneyness \( x = \log \frac{K}{F_T} \) in the forward price \( F_T = S_0e^{rT} \) is given by

\[
w_g(T, x) = a + \frac{r - \ell}{2}(x - m) + \frac{r + \ell}{2} \sqrt{(x - m)^2 + p^2}; \quad (B1)
\]

where \( a, r, \ell, m, p \) are parameters that depend on \( T \). Here, the calibrated parameters are given by

\[
a = 0.01 + 0.03\sqrt{T + 0.04}, \quad r = 0.06(1 - 0.87\sqrt{T}), \\
\ell = 0.31(1 - 0.7\sqrt{T}), \quad m = 0.03 + 0.01T, \\
p = 0.15(0.4 + 0.6\sqrt{T + 0.04}).
\]
B.2 **Constructing the local volatility surface**

In order to compute the local surface, we apply a transformation that corresponds to Dupire’s formula [10] for the SVI parametrization.

\[
    w_{\text{local}}(x, T) = \frac{w_g(T, x) + T \frac{\partial w_g}{\partial T}(T, x)}{\left(1 - \frac{x}{\frac{\partial w_g}{\partial x}}\right)^2 - \left(\frac{\partial w_g}{\partial x}\right)^2 \left((\frac{w_g T}{2} + 1)^2 - 1\right) + \frac{T}{2} \frac{\partial^2 w_g}{\partial x^2}}.
\]

This gives us a local volatility surface in terms of \(T\) and \(x\).

B.3 **Using the local volatility surface**

When we use the local volatility surface, we replace \(K\) by \(s\) and \(T\) by \(t\). In order to see what that implies for the log moneyness, we need to go back to the definition. There we had \(x = \log \frac{K}{F_T}\), then when we use the local volatility surface

\[
    x(s, S_0, t) = \log \frac{s}{F_t(S_0)} = \log \frac{s}{S_0e^{rt}}.
\]

The local volatility is now given by

\[
    \sigma(s, t) = \sqrt{w_{\text{local}}(x(s, S_0, t), t)}.
\]

Note that \(\sigma(s, t)\) cannot be directly evaluated for very small values of \(s\). A function that evaluates this volatility surface can be downloaded from the BENCHOP web site.

**References**


[58] A.I. Tolstykh and D.A. Shirobokov, On using radial basis functions in a “finite difference mode” with applications to elasticity problems, Computational Mechanics 33 (2003), pp. 68–79.


